AF 2676

Practitioner's Docket No. NVIDP069/P000051 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Mark J. Kilgard et al.

Application No.: 10/006,477 Group No.: 2676
Filed: 11/30/2001 Examiner: Quillen, Allen
For: FLOATING POINT BUFFER SYSTEM AND METHOD FOR USE DURING PROGRAMMABLE FRAGMENT PROCESSING IN A GRAPHICS PIPELINE

Mail Stop Appeal Briefs – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION--37 C.F.R. § 1.192)

1. Transmitted herewith, in triplicate, is the APPEAL BRIEF in this application, with respect to the Notice of Appeal filed on March 9, 2004.

2. STATUS OF APPLICANT

This application is on behalf of other than a small entity.

CERTIFICATION UNDER 37 C.F.R. §§ 1.8(a) and 1.10*

*(When using Express Mail, the Express Mail label number is **mandatory;** Express Mail certification is optional.)*

I hereby certify that, on the date shown below, this correspondence is being:

MAILING

X deposited with the United States Postal Service in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

37 C.F.R. § 1.8(a)
_ with sufficient postage as first class mail.

37 C.F.R. § 1.10*
_ as "Express Mail Post Office to Addressee"
Mailing Label No. ______________ **(mandatory)**

TRANSMISSION

_ facsimile transmitted to the Patent and Trademark Office, (703) ______ - ____________.

Melissa D. Orvis
Signature

Date: 4/30/04

Melissa D. Orvis
(type or print name of person certifying)

** Only the date of filing (' 1.6) will be the date used in a patent term adjustment calculation, although the date on any certificate of mailing or transmission under ' 1.8 continues to be taken into account in determining timeliness. See ' 1.703(f). Consider "Express Mail Post Office to Addressee" (' 1.10) or facsimile transmission (' 1.6(d)) for the reply to be accorded the earliest possible filing date for patent term adjustment calculations.*

3. FEE FOR FILING APPEAL BRIEF

Pursuant to 37 C.F.R. § 1.17(c), the fee for filing the Appeal Brief is:

other than a small entity	$330.00

Appeal Brief fee due $330.00

4. EXTENSION OF TERM

The proceedings herein are for a patent application and the provisions of 37 C.F.R. § 1.136 apply.

Applicant believes that no extension of term is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

5. TOTAL FEE DUE

The total fee due is:

Appeal brief fee	$330.00
Extension fee (if any)	$0.00

TOTAL FEE DUE $330.00

6. FEE PAYMENT

Attached is a check in the amount of $330.00.

A duplicate of this transmittal is attached.

7. FEE DEFICIENCY

If any additional extension and/or fee is required, and if any additional fee for claims is required, charge Deposit Account No. 50-1351 (Order No. NVIDP069).

Signature of Practitioner

Reg. No.: 41,429
Tel. No.: 408-971-2573
Customer No.: 28875

Kevin J. Zilka
Silicon Valley IP Group, PC
P.O. Box 721120
San Jose, CA 95172-1120
USA

9 Brief 5/6/04 182

Practitioner's Docket No. NVIDP069/P000051 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Mark J. Kilgard et al.

Application No.: 10/006,477 Group No.: 2676

Filed: November 30, 2001 Examiner: Quillen, Allen E.

For: FLOATING POINT BUFFER SYSTEM AND METHOD FOR USE DURING PROGRAMMABLE FRAGMENT PROCESSING IN A GRAPHICS PIPELINE

Commissioner for Patents
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

APPELLANT'S BRIEF (37 C.F.R. § 1.192)

This brief is in furtherance of the Notice of Appeal, filed in this case on March 9, 2004.

The fees required under § 1.17, and any required petition for extension of time for filing this brief and fees therefor, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate. (37 C.F.R. § 1.192(a))

This brief contains these items under the following headings, and in the order set forth below (37 C.F.R. § 1.192(c)):

I REAL PARTY IN INTEREST
II RELATED APPEALS AND INTERFERENCES
III STATUS OF CLAIMS
IV STATUS OF AMENDMENTS

The final page of this brief bears the practitioner's signature.

I REAL PARTY IN INTEREST (37 C.F.R. § 1.192(c)(1))

The real party in interest in this appeal is NVIDIA Corporation.

II RELATED APPEALS AND INTERFERENCES (37 C.F.R. § 1.192(c)(2))

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no other such appeals or interferences.

III STATUS OF CLAIMS (37 C.F.R. § 1.192(c)(3))

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1, 4-8, and 11-26.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims withdrawn from consideration but not canceled: None
2. Claims pending: 1, 4-8, and 11-26
3. Claims allowed: None

4. Claims rejected: 1, 4-8, and 11-26

C. CLAIMS ON APPEAL

The claims on appeal are: 1, 4-8, and 11-26

IV STATUS OF AMENDMENTS (37 C.F.R. § 1.192(c)(4))

As to the status of any amendment filed subsequent to final rejection, an amendment after final was made on January 26, 2004, but was not entered.

V SUMMARY OF INVENTION (37 C.F.R. § 1.192(c)(5))

As shown in Figure 4 and the accompanying descriptions on pages 16-17, a system, method and computer program product are provided for buffering data in a computer graphics pipeline. Initially, graphics floating point data is read from a buffer in a graphics pipeline. Next, the graphics floating point data is operated upon in the graphics pipeline. Further, the graphics floating point data is stored to the buffer in the graphics pipeline. In another embodiment, the buffer may serve as a texture map.

Optionally, as set forth on pages 9-12, the graphics floating point data may be packed and/or unpacked, whereby a technique is provided for encoding and decoding multiple low-precision data items stored in a single higher-precision data format.

Still yet, a technique is set forth in Figure 4 and the accompanying descriptions on pages 16-17 for buffering data during multi-pass rendering in a computer graphics pipeline. Initially, graphics floating point data are operated on during a rendering pass in a graphics pipeline. During this operation, graphics floating point data may be read from a buffer. Further, the graphics floating point results produced by the rendering

pass are stored to the buffer. During use, the foregoing operations may be repeated during additional rendering passes.

VI ISSUES (37 C.F.R. § 1.192(c)(6))

Issue # 1: The Examiner has rejected Claims 1, 4-8, and 11-26 under 35 U.S.C. 103(a) as being unpatentable over Johnson, U.S. Patent 6,469,704, in view of Deering, Michael F., and Nelson, Scott R., Leo: A System for Cost Effective 3D Shaded Graphics, in further view of Fowler, et al., U.S. Patent Application Publication US 2002/0180741.

VII GROUPING OF CLAIMS (37 C.F.R. § 1.192(c)(7))

The claims of the above groups do not stand or fall together. Following is the grouping of claims. In the following section, appellant explains why the claims of each group are believed to be separately patentable.

Issue # 1: Grouping of Claims –

Group #1: Claims 1-6, 8, 11-13, 15-17, and 26;
Group #2: Claim 7, 14, and 18-19;
Group #3: Claim 20-21;
Group #4: Claims 22-23;
Group #5: Claim 24; and
Group #6: Claim 25.

VIII ARGUMENTS (37 C.F.R. § 1.192(c)(8))

Issue #1:

The Examiner has rejected Claims 1, 4-8, and 11-26 under 35 U.S.C. 103(a) as being unpatentable over Johnson, U.S. Patent 6,469,704, in view of Deering, Michael F., and Nelson, Scott R., Leo: A System for Cost Effective 3D Shaded Graphics, in further view of Fowler, et al., U.S. Patent Application Publication US 2002/0180741. Appellant respectfully disagrees with such rejection.

Group #1: Claims 1-6, 8, 11-13, 15-17, and 26

With respect to the first group of claims, appellant has argued that the Examiner's Johnson-Deering combination fails to meet appellant's specifically claimed "graphics floating point data [which] includes fragment data received from a rasterizer that is ... stored in an unclamped format" (emphasis added), where the "unclamped format [is] dictated by a graphics application program interface" (emphasis added).

In response, the Examiner has asserted the following response which takes a "piecemeal" approach to the application of the claim elements to the prior art. As will be set forth, such rejection is, by itself, improper and, more importantly, still fails to meet all of applicant's claim limitations mentioned above.

Regarding the "fragment data received from a rasterizer" (emphasis added) claim language, in his "Response to Arguments" under final, the Examiner argues that: (1) Appellant discloses that OpenGL provides support for (and mandates) per-fragment operations; (2) Johnson discloses OpenGL, portions of the primitives, rendering; and (3) Fowler teaches congruity of these features.

The Examiner goes on in his "Response to Arguments" to argue that Deering teaches appellant's claimed "graphics floating point data ... that is ... stored in an unclamped format" (emphasis added).

Finally, the Examiner relies on both Johnson and Fowler to make a prior art showing of applicant's claimed "format [that is] dictated by a graphics application program interface."

It appears that the Examiner has simply broken down appellant's claim language into components (i.e. phrases, adjectives, nouns, etc.), and has then attempted to make a "piece-meal" prior art showing of such components. Thus, the Examiner's rejection may be considered analogous to gleaning phrases, adjectives, nouns, etc. from appellant's claims, and then using prior art references collectively as a dictionary to make a prior art showing of such phrases, adjectives, nouns, etc.

A claim is anticipated only if each and every element as set forth in the claim is found, and the elements must be arranged as required by the claim. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). In view of the manner in which the Examiner has simply mapped phrases, adjectives, nouns, etc. of appellant's claims to the prior art, appellant asserts that the elements of the Examiner's proposed combination are simply not arranged as required by appellant's claims.

More importantly, the foregoing rejection still fails to meet all of applicant's claim limitations. Specifically, with respect to the claim language "format [that is] dictated by a graphics application program interface," the Examiner states, on page 4, second paragraph of the Final Office Action mailed 11/25/03, that Johnson discloses applicant's claimed "from a rasterizer in a format dictated by [API]" and cites the following excerpt:

> "Primitive accumulator 428 analyzes the sequentially generated primitive data sets in original graphics call sequence 401 to make the necessary determinations described below to create the primitive command set. In the illustrative embodiment, primitive command set generator 424 includes a graphics call buffer 426 that receives and stores vertex-related graphics calls 405 forming one or more primitive data sets." (col. 13, lines 53-62)

This excerpt, however, relates to vertex processing, and not any sort of "from a rasterizer in a format dictated by [API]," as claimed by applicant. Thus, the Examiner's assertion is simply incorrect, and Johnson does not teach, disclose or even suggest "from a rasterizer in a format dictated by [API]," as claimed.

The Examiner goes on in page 4 of the Final Office Action to admit that "Johnson does not disclose wherein the graphics data includes fragment data received from a rasterizer in a format dictated by [API]." Moreover, the Examiner continues by stating that "Fowler teaches where the graphics data includes fragment data received from a rasterizer in a format dictated by a graphics application program interface [API]." Further, the Examiner cites page 2, paragraphs 29-30, and 39 from Fowler. Thus, it is clear that the Examiner is attempting to rely on the excerpt below to meet applicant's claimed "unclamped format [that is] <u>dictated by a graphics application program interface</u>" (emphasis added).

> "[0029] Within a pixel pipeline, operations performed on the various values of a fragment may proceed along different datapaths and/or at different rates. In order to provide a larger and more detailed texture image, for example, texture maps are generally stored off-chip (e.g. in system memory), and as a consequence the storage access operation required to retrieve a texel value may have a latency of many processing cycles. In some implementations, a texture map may also be compressed for efficient storage and/or transfer, requiring a decompression operation (and resulting in an additional delay) upon retrieval. Such latencies may slow the rate of the texture datapath in relation to the datapaths of other fragment value operations.
>
> [0030] In order to synchronize the presentation of the various fragment values to the pixel combiner, it may be desirable to buffer one datapath to account for a delay in another datapath. FIG. 6 shows a block diagram of a 3D architecture having a pipeline 134 that includes a FIFO buffer 180 in a pass-through datapath. In an exemplary application, color and/or location values are carried on the pass-through datapath, and FIFO 180 compensates for latencies encountered in a texture datapath.

> [0039] One bump-mapping technique that is supported by the Direct3D and OpenGL APIs is environment-mapped bump mapping (EMBM). FIGS. 8A and 8B show how EMBM may be implemented in a serial fashion. In FIG. 8A, TL&F 150 uses a coordinate pair (e.g. in the ST texture coordinate space) to reference a specialized texture map called a `bump map ` (also called a `perturbation map,` `texture coordinate displacement map,` or simply `displacement map`). Instead of a texel value, the referenced map location contains a displacement vector (ds, dt) that TL&F 150 applies to perturb the coordinate pair (or, alternatively, another set of coordinate values) to obtain a new coordinate pair in another coordinate space (e.g. an S`T` environment coordinate space). For example, a TL&F operating in response to commands from a Direct3D API may apply a coordinate perturbation according to the following matrix equation: 1 [s ' t '] = [s t] + ([ds dt] x [M 00 M 01 M 10 M 11])"

This excerpt, however, along with the remaining Fowler reference is significantly lacking. For example, there is simply no mention in the foregoing except of any sort of "format," let alone an "unclamped format [that is] <u>dictated by a graphics application program interface</u>" (emphasis added). Fowler's Direct3D API matrix-based coordinate perturbation simply does not meet appellant's "format," as specifically claimed.

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on appellant's disclosure. *In re Vaeck,*947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

In view of the foregoing deficiencies, appellant contends that the third element of the *prima facie* case of obviousness has simply not been met. Specifically, the prior art references when combined fail to teach or suggest all the claim limitations embodied in

the claim language: "graphics floating point data [which] includes fragment data received from a rasterizer that is … stored in an unclamped format" (emphasis added), where the "unclamped format [is] dictated by a graphics application program interface" (emphasis added). Again, just by way of example, the Examiner has failed to make a prior art showing of appellant's claimed "unclamped format [that is] dictated by a graphics application program interface" (emphasis added), in the specific context of the remaining limitations.

With respect to the first element of the *prima facie* case of obviousness, the Examiner contends that there is suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to combine the teachings of the references. In view of the vast evidence to the contrary, appellant respectfully disagrees.

For example, Deering's vertex processing *teaches away* from the *non-analogous* technique of storing fragment data received from a rasterizer in an unclamped format. In particular, the excerpts from Deering and Johnson relied upon by the Examiner relate to vertex processing, while Fowler relates to fragment processing. To simply glean features from the vertex processing arts, such as Deering and Johnson, and combine the same with the *non-analogous art* of fragment processing, such as that of Fowler, would simply be improper. Vertex processing relates to the problem of transforming vertices while fragment processing relates to the problem of shading pixels.

"In order to rely on a reference as a basis for rejection of an applicant's invention, the reference must either be in the field of applicant's endeavor or, if not, then be reasonably pertinent to the particular problem with which the inventor was concerned." In re Oetiker, 977 F.2d 1443, 1446, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992). See also In re Deminski, 796 F.2d 436, 230 USPQ 313 (Fed. Cir. 1986); In re Clay, 966 F.2d 656, 659, 23 USPQ2d 1058, 1060-61 (Fed. Cir. 1992) In view of the vastly different

types of problems addressed during vertex processing as opposed to fragment processing, the Examiner's proposed combination is inappropriate.

Further, applicant's claimed fragment-related features would have been unobvious in view of Deering and Johnson, since Deering and Johnson's vertex processing *teaches away* from any sort of fragment processing. *In re Hedges*, 783 F.2d 1038, 228 USPQ 685 (Fed. Cir. 1986).

Thus, only appellant teaches and claims the *unobvious* technique of fragment data processing in an unclamped format, as specifically claimed. To date, the Examiner has not addressed appellant's previously submitted arguments regarding this issue.

As set forth in the originally filed specification, most computations dealing with fragment data are typically constrained to operate on values in the range [0,1]. Computational results are also typically clamped to the range [0,1]. Color, texture, and depth buffers themselves also hold values mapped to the range [0, 1]. Unfortunately, these constraints and the limited precision of typical computations can result in reduced accuracy during fragment processing.

Only appellant recognizes the foregoing limitations of the prior art, and teaches a combination of features for overcoming the foregoing problem.

Group #2: Claim 7, 14, and 18-19

Regarding appellant's claimed "wherein the buffer serves as a texture map," the Examiner has merely pointed to a mention of a "texture map" at the designated locations in the text of Deering. Appellant asserts that this is insufficient to make a showing of a "buffer," that serves as a texture map, as claimed. Appellant contends that

the third element of the *prima facie* case of obviousness has not been met with respect to the foregoing limitations.

Group #3: Claim 20-21

Regarding appellant's claimed "repeating (a) – (c) during additional rendering passes utilizing results of a previous rendering pass," where (a) – (c) include: "operating on graphics floating point data during a rendering pass in a graphics pipeline … reading the graphics floating point data from a buffer during the rendering pass … [and] storing the graphics floating point data to the buffer during the rendering pass," the Examiner has relied on the "coalescing primitive data sets" shown in Figure 5B and described in col. 15, line 60 – col. 16, line 67. Appellant has carefully reviewed such excerpts, and there is simply no disclosure of "repeating (a) – (c) during additional rendering passes <u>utilizing results of a previous rendering pass</u>" (emphasis added), in the specifically claimed context. Thus, appellant again contends that the third element of the *prima facie* case of obviousness has not been met with respect to the foregoing limitations.

Group #4: Claims 22-23

With respect to appellant's claimed "packing the graphics floating point data in the graphics pipeline … and storing the graphics floating point data to a buffer; wherein the packing facilitates storage of at least two quantities in a single buffer in a single pass," the Examiner has relied on Figures 6 and 10, and the following excerpts from Fowler.

> "[0029] Within a pixel pipeline, operations performed on the various values of a fragment may proceed along different datapaths and/or at different rates. In order to provide a larger and more detailed texture image, for example, texture maps are generally stored off-chip (e.g. in system memory), and as a consequence the storage access operation required to retrieve a texel value may have a latency of many processing cycles. In some implementations, a texture map may also be compressed for efficient storage and/or transfer, requiring a

> decompression operation (and resulting in an additional delay) upon retrieval. Such latencies may slow the rate of the texture datapath in relation to the datapaths of other fragment value operations.
>
> [0030] In order to synchronize the presentation of the various fragment values to the pixel combiner, it may be desirable to buffer one datapath to account for a delay in another datapath. FIG. 6 shows a block diagram of a 3D architecture having a pipeline 134 that includes a FIFO buffer 180 in a pass-through datapath. In an exemplary application, color and/or location values are carried on the pass-through datapath, and FIFO 180 compensates for latencies encountered in a texture datapath.
>
> [0046] It is possible that a multipass operation such as EMBM may be used to render only a portion of the pixels of a frame (i.e. fewer than all of the objects in the rendering space). Therefore, it may be desirable for a pipeline 138 as shown in FIG. 10 to support both single-pass and multipass operations. For example, the pipeline may include one or more multiplexers (or a similar mechanism) that may be controlled to configure the texture datapath as appropriate."

After careful review of such excerpts, appellant has confirmed that there is simply no disclosure of the foregoing claimed feature involving any sort of <u>packing</u> of "graphics floating point data," where the packing facilitates storage of <u>at least two quantities in a single buffer in a single pass</u>, as claimed.

Thus, appellant again contends that the third element of the *prima facie* case of obviousness has not been met with respect to the foregoing limitations.

Group #5: Claim 24

With respect to appellant's claimed "determining whether the graphics pipeline is operating in a programmable mode utilizing a command associated with a graphics application program interface … and <u>if it is determined that the graphics pipeline is operating in the programmable mode</u>, storing the graphics floating point data to a frame buffer," the Examiner has relied on Figure 6 and paragraphs 29-30 (note above) from Fowler.

The Examiner has admitted that Johnson does not disclose the foregoing emphasized subject matter, and that Fowler is relied upon for a prior art showing. See page 5, last full paragraph in the Final Office Action mailed 11/25/03. Fowler, however, does not disclose, teach or even suggest any sort of conditional storage of graphics floating point data to a frame buffer, where the condition is "<u>if it is determined that the graphics pipeline is operating in the programmable mode</u>," as claimed by appellant.

Moreover, it is noted that the arguments applicable to Group #1, are equally applicable here. Therefore, appellant again contends that the third element of the *prima facie* case of obviousness has not been met with respect to the foregoing limitations.

Group #5: Claim 25

Appellant has previously emphasized that Claim 25 requires that "the buffer serves as a texture map by using previous rendering results via an extension of an application program interface," and requested a specific prior art showing of such limitations or a notice of allowance.

After carefully reviewing page 5 of the Examiner's Final Office Action mailed 11/25/03, where Claim 25 is addressed, however, appellant can find no specific consideration of appellant's claimed "buffer [that] serves as a texture map <u>by using previous rendering results via an extension of an application program interface</u>." Thus, appellant contends that the third element of the *prima facie* case of obviousness has not been met with respect to Claim 25.

In view of the remarks set forth hereinabove, all of the independent claims are deemed allowable, along with any claims depending therefrom.

IX APPENDIX OF CLAIMS (37 C.F.R. § 1.192(c)(9))

The text of the claims involved in the appeal is:

1. (Previously Amended) A method for buffering data produced by a computer graphics pipeline, comprising:
producing graphics floating point data in a graphics pipeline;
operating on the graphics floating point data in the graphics pipeline; and
storing the graphics floating point data to a buffer;
wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data.

2.-3. (Cancelled)

4. (Previously Amended) The method as recited in claim 12, wherein the fragment data includes color data.

5. (Previously Amended) The method as recited in claim 12, wherein the fragment data includes depth data.

6. (Original) The method as recited in claim 1, wherein the graphics floating point data is only constrained by an underlying data type.

7. (Original) The method as recited in claim 1, wherein the buffer serves as a texture map.

8. (Previously Amended) A computer program product for buffering data produced by a computer graphics pipeline, comprising:

(a) computer code for producing graphics floating point data in a graphics pipeline;

(b) computer code for operating on the graphics floating point data in the graphics pipeline; and

(c) computer code for storing the graphics floating point data to a buffer;

(d) wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data.

9.-10. (Canceled)

11. (Previously Amended) The computer program product as recited in claim 8, wherein the fragment data includes color data.

12. (Previously Amended) The computer program product as recited in claim 8, wherein the fragment data includes depth data.

13. (Original) The computer program product as recited in claim 8, wherein the graphics floating point data is only constrained by an underlying data type.

14. (Original) The computer program product as recited in claim 8, wherein the buffer serves as a texture map.

15. (Previously Amended) A system for buffering data produced by a computer graphics pipeline, comprising:

(a) logic for producing graphics floating point data in a graphics pipeline;

(b) logic for operating on the graphics floating point data in the graphics pipeline; and

(c) logic for storing the graphics floating point data to a buffer;

(d) wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data.

16. (Previously Amended) A buffering apparatus, comprising:

(a) a buffer capable of storing graphics floating point data produced by a graphics pipeline;

(b) wherein the graphics floating point data includes fragment data received from a rasterizer that is stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data.

17. (Previously Amended) A system for buffering data produced by a computer graphics pipeline, comprising:

(a) means for producing graphics floating point data in a graphics pipeline;

(b) means for operating on the graphics floating point data in the graphics pipeline; and

(c) means for storing the graphics floating point data to a buffer;

(d) wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data.

18. (Previously Amended) A method for buffering data produced by a computer graphics pipeline, comprising:

(a) producing graphics floating point data in a graphics pipeline;

(b) operating on the graphics floating point data in the graphics pipeline; and

(c) storing the graphics floating point data to a buffer;

(d) wherein the buffer serves as a texture map.

19. (Previously Amended) A buffering apparatus, comprising:

(a) a buffer capable of storing graphics floating point data produced by a graphics pipeline;

(b) wherein the buffer serves as a texture map.

20. (Previously Amended) A method for buffering data during multi-pass rendering, comprising:

(a) operating on graphics floating point data during a rendering pass in a graphics pipeline;

(b) reading the graphics floating point data from a buffer during the rendering pass;

(c) storing the graphics floating point data to the buffer during the rendering pass; and

(d) repeating (a) – (c) during additional rendering passes utilizing results of a previous rendering pass.

21. (Original)The method as recited in claim 20, wherein the operating includes deferred shading.

22. (Previously Amended) A method for buffering data produced by a computer graphics pipeline, comprising:
producing graphics floating point data in a graphics pipeline;
packing the graphics floating point data in the graphics pipeline; and
storing the graphics floating point data to a buffer;
wherein the packing facilitates storage of at least two quantities in a single buffer in a single pass.

23. (Previously Amended) A method for buffering data produced by a computer graphics pipeline, comprising:
producing graphics floating point data in a graphics pipeline;
unpacking the graphics floating point data in the graphics pipeline; and
operating on the unpacked graphics floating point data in the graphics pipeline;
wherein the unpacking facilitates storage of at least two quantities in a single buffer in a single pass.

24. (Previously Amended) A method for buffering data produced by a computer graphics pipeline, comprising:
operating on graphics floating point data in a graphics pipeline;
producing the graphics floating point data in the graphics pipeline;
determining whether the graphics pipeline is operating in a programmable mode utilizing a command associated with a graphics application program interface;
if it is determined that the graphics pipeline is not operating in the programmable mode, performing standard graphics application program interface operations on the graphics floating point data; and
if it is determined that the graphics pipeline is operating in the programmable mode:
storing the graphics floating point data to a frame buffer,

wherein the graphics floating point data includes fragment data received from a rasterizer that is read and stored in an unclamped format dictated by a graphics application program interface extension for increasing a parameter selected from the group consisting of a precision and a range of the graphics floating point data.

25. (Previously Presented) The buffering apparatus as recited in claim 19, wherein the buffer serves as the texture map by using previous rendering results via an extension of an application program interface.

26. (Previously Presented) The method as recited in claim 1, wherein the buffer includes a frame buffer.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 971-2573. For payment of any additional fees due in connection with the filing of this paper, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1351 (Order No. NVIDP069_P000051).

Respectfully submitted,

By: ______________________ Date: 04/30/04

Kevin J. Zilka
Reg. No. 41,429

Silicon Valley IP Group, P.C.
P.O. Box 721120
San Jose, California 95172-1120
Telephone: (408) 971-2573
Facsimile: (408) 971-4660